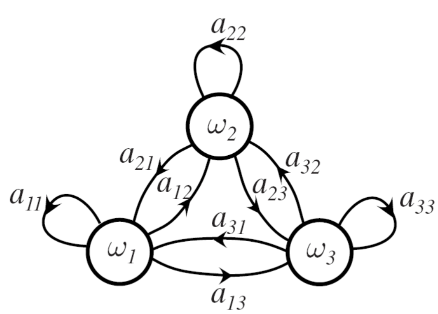
CSE6328.3
Speech & Language Processing

No.6

Hidden Markov Model (HMM)

Prof. Hui Jiang
Department of Computer Science and Engineering
York University

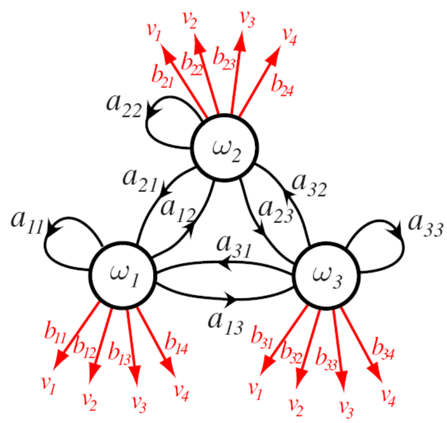
Markov Chain Model: review



- Containing a set of states
- Probability of observing a state depends on its immediate history
- 1st-order Markov chain: history ➔ previous state
  - Characterized by a transition matrix $\{a_{ij}\}$ and an initial prob vector
- ***Directly observing a sequence of states:***

  $X = \{\omega_1, \omega_4, \omega_2, \omega_2, \omega_1, \omega_4\}$
- $Pr(X) = P(\omega_1)\, a_{14}\, a_{42}\, a_{22}\, a_{21}\, a_{14}$

Hidden Markov Model (HMM)

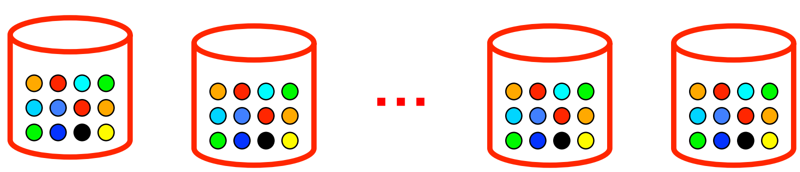


S = $\omega_1, \omega_3, \omega_2, \omega_2, \omega_1, \omega_3$ **(hidden)**

O = v4, v1, v1, v4, v2, v3 **(observed)**

- **HMM is also called a probabilistic function of a Markov chain**
  - **State transition follows a Markov chain.**
  - **In each state, it generates observation symbols based on a probability function. Each state has its own prob function.**
  - **HMM is a doubly embedded stochastic process.**
- **In HMM,**
  - **State is not directly observable (hidden states)**
  - **Only observe observation symbols generated from states**

HMM example: Urn & Ball



Urn 1	Urn 2	…	Urn N-1	Urn N
Pr(RED) = $b_1(1)$	Pr(RED) = $b_2(1)$		Pr(RED) = $b_{N-1}(1)$	Pr(RED) = $b_N(1)$
Pr(BLE) = $b_1(2)$	Pr(BLE) = $b_2(2)$		Pr(BLE) = $b_{N-1}(2)$	Pr(BLE) = $b_N(2)$
Pr(GRN) = $b_1(3)$	Pr(GRN) = $b_2(3)$		Pr(GRN) = $b_{N-1}(3)$	Pr(GRN) = $b_N(3)$
…	…		…	…

Observation: O = { GRN, GRN, BLE, RED, RED, … BLE}

Elements of an HMM

- An HMM is characterized by the following:
  - *N*: the number of states in the model
  - *M*: the number of distinct observation symbols
  - *A = {a_{ij}} (1<=i,j<=N)*: the state transition probability distribution, called ***transition matrix.***

$$a_{ij} = \Pr(q_t = S_j \mid q_{t-1} = S_i) \qquad (1 \le i, j \le N)$$

  - *B={$b_j(k)$} (1<=j<=N, 1<=k<=M):* observation symbol probability distribution in all states.

$$b_j(k) = \Pr(v_k \mid q_t = S_j) \qquad (1 \le j \le N, 1 \le k \le M)$$

  - π_i (1<=i<=N): initial state distribution
- The complete parameter set of an HMM is denoted as $\Lambda = \{A, B, \pi\}$, where *A* is transition matrix, *B* is observation functions, π is initial probability vector.

An HMM process

- Given an HMM, denoted as $\Lambda = \{A, B, \pi\}$ and an observation sequence *O={O_1,O_2, …, O_T}.*
- The HMM can be viewed as a generator to produce *O* as:
  1. Choose an initial state $q_1=S_i$ according to the initial probability distribution π.
  2. Set *t=1*.
  3. Choose an observation O_t according to the symbol observation probability distribution in state S_i, i.e., $b_i(k)$.
  4. Transit to a new state $q_{t+1} = S_j$ according to the state transition probability distribution, i.e., a_{ij}.
  5. Set *t=t+1*, return to step 3 if *t<T*.
  6. Terminate the procedure.

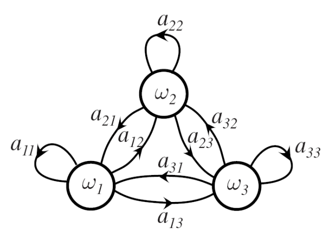
Basic Assumptions in HMM

- Markov Assumption:
  - State transition follows a 1st-order Markov chain.
  - This assumption implies the duration in each state, *j*, is a geometric distribution: $p_j(d) = (a_{jj})^{d-1}(1-a_{jj})$
- Output Independence Assumption: the probability that a particular observation symbol is emitted from HMM at time *t* depends only on the current state s_t and is conditionally independent of the past and future observations.
- The two assumptions limit the memory of an HMM and may lead to model deficiency. But they significantly simplify HMM computation, also greatly reduce the number of free parameters to be estimated in practice.
  - Some research works to relax these assumptions has been done in the literature to enhance HMM in modeling speech signals.

Types of HMMs (I)

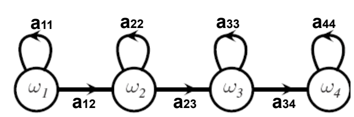
- Different transition matrices:
  - Ergodic HMM Topology: (with full transition matrix)

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$



  - Left-to-right HMM Topology: states proceed from left to right

$$A = \begin{bmatrix} a_{11} & a_{12} & 0 & 0 \\ 0 & a_{22} & a_{23} & 0 \\ 0 & 0 & a_{33} & a_{34} \\ 0 & 0 & 0 & a_{44} \end{bmatrix}$$



Types of HMMs (II)

- Different observation symbols: discrete vs. continuous
  - Discrete density HMM (DDHMM): observation is discrete, one of a finite set. In discrete density HMM, observation function is a discrete probability density, i.e., a table. In state *j,*

$$\begin{array}{c} \\ O_j(k) = \end{array} \begin{array}{cccc} v_1 & v_2 & v_3 & v_4 \\ [0.1 & 0.4 & 0.3 & 0.2] \end{array}$$

  - Continuous density HMM (CDHMM): observation *x* is continuous in an observation space. In CDHMM, observation function is a probability density function (p.d.f.). The common function forms:
    - Multivariate Gaussian distribution

$$p_j(\mathbf{x}) = N(\mathbf{x} \mid \boldsymbol{\mu}_j, \Sigma_j) = \frac{1}{\sqrt{(2\pi)^n \mid \Sigma_j \mid}} e^{-(\mathbf{x}-\boldsymbol{\mu}_j)^t \Sigma_j^{-1} (\mathbf{x}-\boldsymbol{\mu}_j)/2} \quad -\infty < \mathbf{x} < \infty$$

    - Gaussian mixture model

$$p_j(x) = \sum_{k=1}^{K} \omega_{ik} N(x \mid \boldsymbol{\mu}_{ik}, \Sigma_{ik}) \quad \sum_{k=1}^{k} \omega_k = 1 \quad 0 < \omega_m < 1 \quad K > 1$$

HMM for data modeling(I)

- HMM is used as a powerful statistical model for sequential and temporary data observation.
- HMM is theoretically (mathematically) sound; relatively simple learning and decoding algorithms exists.
- HMM is widely used in pattern recognition, machine learning, etc.
  - Speech recognition: model speech signals.
  - Statistical language processing: model language (word/semantics sequence).
  - OCR (optimal character recognition): model 2-d character image.
  - Gene finding: model DNA sequence (profile HMM),

HMM for data modeling(II)

- How to use HMM to model sequential data ?
  - The entire data sequence is viewed as one data sample O.
  - The HMM is characterized by its parameters $\Lambda = \{A, B, \pi\}$.
- Learning Problem: HMM parameters Λ must be estimated from a data sample set {O_1,O_2, …, O_T}.
  - The HMM parameters are set so as to best explain known data.
  - "Best" in different senses:
    - Maximum likelihood estimation.
    - Maximum a posteriori (MAP) estimation.
    - Discriminative training: minimum classification error (MCE) in training data, Maximum mutual information (MMI) estimation.
- Evaluation Problem: for an unknown data sample Ox, calculate the probability of the data sample given the model, p(Ox|Λ).
- Decoding Problem: uncover the hidden information; for an observation sequence O={o_1,o_2,…,o_t}, decode the best state sequence Q={s_1,s_2,…,s_t} which is optimal in explaining O.

HMM Computation(1): Evaluation(I)

- Given a known HMM $\Lambda = \{A, B, \pi\}$, how to compute the probability of an observation data *O={o1,o2,…,oT}* generated by the HMM, i.e., *p(O|Λ)*.
- Direct computation: In HMM, the observation data O can be generated by any a valid state sequence (with length *T*) with different probability. The probability of *O* generated by the whole model is the summation of all these probabilities. Assume *S={s1,s2,…,sT}* is a valid state sequence in HMM,

$$p(O \mid \Lambda) = \sum_S p(O, S \mid \Lambda) = \sum_S p(S \mid \Lambda) \cdot p(O \mid S, \Lambda)$$

$$= \sum_{s_1 \cdots s_T} \left[p(s_1 \mid \Lambda) \cdot \prod_{t=2}^{T} p(s_t \mid s_{t-1}, \Lambda) \cdot \prod_{t=1}^{T} p(o_t \mid s_t, \Lambda) \right]$$

$$= \sum_{s_1 \cdots s_T} \left[\pi_{s_1} \cdot \prod_{t=2}^{T} a_{s_{t-1} s_t} \cdot \prod_{t=1}^{T} b_{s_t}(o_t) \right]$$

HMM Computation(1): Evaluation(II)

- For Gaussian mixture CDHMM,

$$p(O\mid\Lambda) = \sum_{s_1\cdots s_T}\left[\pi_{s_1}\cdot\prod_{t=2}^{T}a_{s_{t-1}s_t}\cdot\prod_{t=1}^{T}b_{s_t}(o_t)\right]$$

$$= \sum_{s_1\cdots s_T}\left[\pi_{s_1}\cdot\prod_{t=2}^{T}a_{s_{t-1}s_t}\cdot\prod_{t=1}^{T}\sum_{k=1}^{K}N(o_t\mid\mu_{s_tk},\Sigma_{s_tk})\right]$$

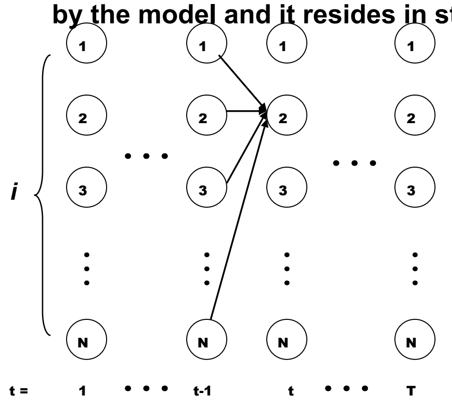
$$= \sum_{s_1\cdots s_T}\sum_{l_1\cdots l_T}\left[\pi_{s_1}\cdot\prod_{t=2}^{T}a_{s_{t-1}s_t}\cdot\prod_{t=1}^{T}N(o_t\mid\mu_{s_tl_t},\Sigma_{s_tl_t})\right]$$

where *$l=\{l_1,\ldots,l_T\}$* is the mixture component label sequence. *l_t ($1\le l_t \le K$)* is l_t-th Gaussian mixand in s_t-th HMM state.

- However, the above direct calculation is computationally prohibitive. Even for DDHMM, it is on the order of $O(2T\cdot N^T)$.
  - For N=5, T=100, computation on the order of $2\times100\times5^{100}\approx10^{72}$.
- Obviously, we need an efficient way to calculate *p(O|Λ)*.

HMM Computation(1): Forward-Backward algorithm(I)

- Solution: calculate *p(O|Λ)* recursively.
- Define forward probability: $\alpha_i(t)=\Pr(o_1,o_2,\cdots o_t,q_t=s_i\mid\Lambda)$, the probability of the partial observation sequence (until *t*) generated by the model and it resides in state s_i at time *t*.



forward probabilities

$$\alpha_j(t)=\begin{cases}\pi_j\cdot b_j(o_1) & t=1\\ \sum_{i=1}^{N}\alpha_i(t-1)a_{ij}b_j(o_t) & t>1\end{cases}$$

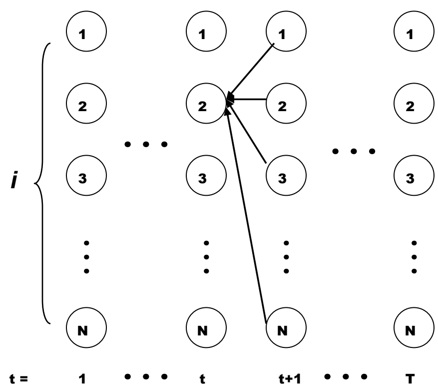
Obviously

$$p(O\mid\Lambda)=\sum_{j=1}^{N}\alpha_j(T)$$

Computational complexity is on the order of $O(N^2T)$.

HMM Computation(1): Forward-Backward algorithm(II)

- Similarly define backward probability: $\beta_i(t) = \Pr(o_{t+1}, o_{t+2}, \cdots o_T \mid q_t = s_i, \Lambda)$
  the probability of generating the partial observation sequence from t+1 to the end by the model, and it resides in state s_i at time *t*.



backward probabilities

$$\beta_i(t) = \begin{cases} 1 & t = T, 1 \le i \le N \\ \sum_{j=1}^{N} \beta_j(t+1) a_{ij} b_j(o_{t+1}) & \text{otherwise} \end{cases}$$

Obviously

$$p(O \mid \Lambda) = \sum_{j=1}^{N} \pi_j \cdot \beta_j(1) \cdot b_j(o_1)$$

Computational complexity is on the order of $O(N^2 T)$.

HMM Computation(1): Forward-Backward algorithm(III)

- If we calculate all forward and backward probabilities:

$$\alpha_i(t) = \Pr(o_1, o_2, \cdots o_t, q_t = s_i \mid \Lambda)$$

$$\beta_i(t) = \Pr(o_{t+1}, o_{t+2}, \cdots o_T \mid q_t = s_i, \Lambda)$$

For any time t, we have

$$p(O \mid \Lambda) = \sum_{i=1}^{N} \alpha_i(t) \cdot \beta_i(t) \qquad \text{(for any } t\text{)}$$

HMM Computation(2): HMM Decoding

- Given a known HMM $\Lambda = \{A, B, \pi\}$ and an observation data sequence $O=\{o_1,o_2,\ldots,o_T\}$, how to find the optimal state sequence associated with the given observation sequence O?
- Optimal in what sense??
  - Could be locally optimal. For any time instant t, find a single best state s_t ➔ generate a path from s_1 to s_T.
  - Prefer a global optimization ➔ find a single best state sequence (also called a path in HMM), which is optimal as a whole.

$$S^* = \arg\max_S p(S \mid O, \Lambda) = \arg\max_S p(S, O \mid \Lambda)$$

$$S^* = \{s_1^*, s_2^*, \cdots, s_T^*\} = \arg\max_{s_1,\cdots,s_T} p(s_1,\cdots,s_T, o_1, \cdots o_T \mid \Lambda)$$

  - Viterbi algorithm: find the above optimal path efficiently.

Viterbi Decoding Algorithm (I)

- Define Optimal Partial Path Score

$$\delta_i(t) = \max_{s_1^{t-1}} P(s_1^{t-1}, s_t = i, O_1^t \mid \Lambda)$$

1. Initialization $\delta_i(0) = \pi_i$
2. DP-Recursion and Bookkeeping

$$\delta_j(t) = \max_{1\le i\le N} [\delta_i(t-1) a_{ij}] b_j(o_t) \quad 1 \le t \le T \quad 1 \le j \le N$$

$$\psi_j(t) = \arg\max_{1\le i\le N} [\delta_i(t-1) a_{ij}] \quad 1 \le t \le T \quad 1 \le j \le N$$

3. Termination

$$P_{\max} = \max_S p(S, O \mid \Lambda) = \max_{1\le i\le N} \delta_i(T) \quad \text{and} \quad \hat{s}_T = \arg\max_{1\le j\le N} \psi_j(T)$$

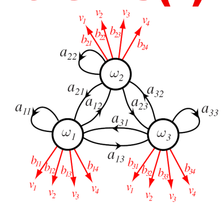
4. Path backtracking

$$\hat{s}_{t-1} = \psi_{\hat{s}_t}(t) \quad \mathrm{t = T, T\text{-}1, \ldots, 2}$$

5. “Optimal” State Sequence:

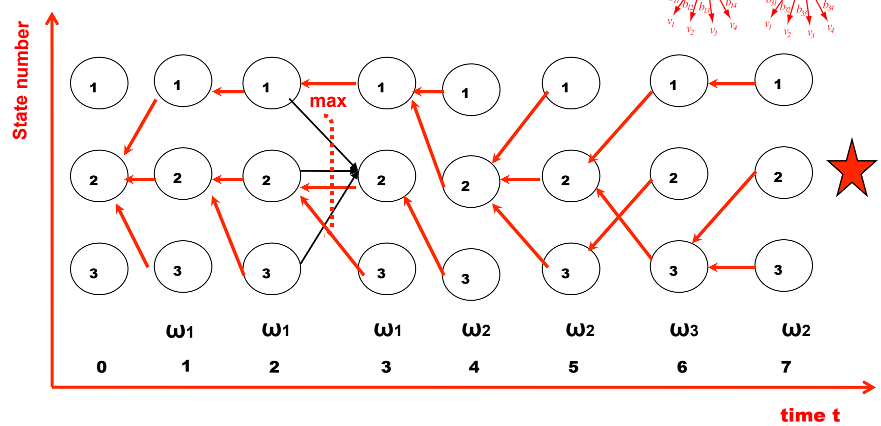
$$\hat{\mathrm{s}} = (\hat{s}_1, \ldots, \hat{s}_T)$$

Viterbi Decoding Algorithm: trellis(I)



Example 1: 3-state ergodic HMM

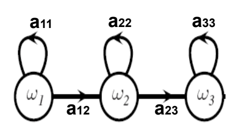
For an observation O={o1,o2,o3,o4,o5,o6,o7}

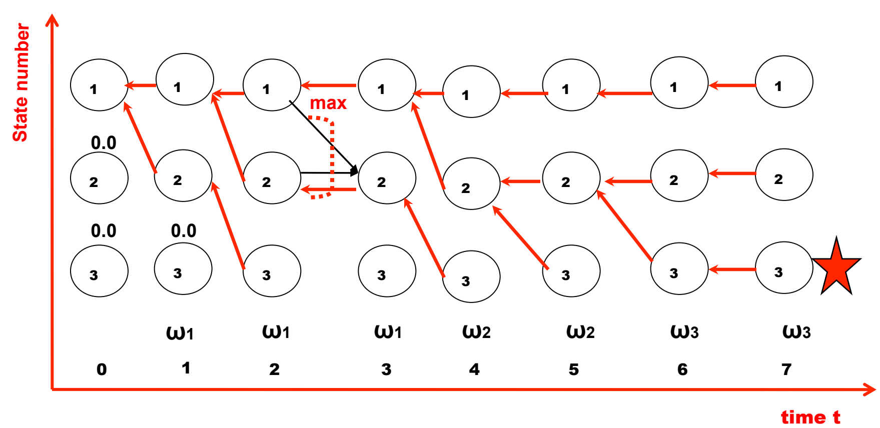


Viterbi Decoding Algorithm: trellis(II)

Example 2: 3-state left-right HMM

For an observation O={o1,o2,o3,o4,o5,o6,o7}





HMM Computation(3): Estimation

- In practice, usually manually select the topology of HMM, including number of states, number of mixture per state, etc.
- However, other HMM parameters must be estimated from training data. (called HMM training)
- HMM training (estimation) criteria:
  - Maximum Likelihood estimation (MLE): maximize the likelihood function of the given training data; HMM parameters are chosen to best reflect the observed data.
  - Maximum a posteriori (MAP) estimation: tune HMM to reflect data as well as some prior knowledge; optimally combine some prior knowledge with data.
  - Discriminative training: increase the discriminative power of all different HMMs (e.g., each HMM for one class); not only adjust HMMs to reflect data, but also try to make all different HMMs as dissimilar as possible.
    - MMIE (maximum mutual information estimation)
    - MCE (minimum classification error) estimation

ML estimation of HMM: Baum-Welch method

- HMM parameters include: $\Lambda = \{A, B, \pi\}$
- Given a set of observation data from this HMM, e.g.
  $D = \{O_1, O_2, \ldots, O_L\}$, each data O_l is a sequence presumably generated by the HMM
- Maximum Likelihood estimation: adjust HMM parameters to maximize the probability of observation set D: $\Lambda = \{A, B, \pi\}$

$$\Lambda_{ML} = \arg\max_{\Lambda} \; p(D \mid \Lambda) = \arg\max_{\Lambda} \; p(O_1, O_2, \cdots, O_L \mid \Lambda)$$

- Similar to GMM, no simple solution exists.
- Baum-Welch method: iterative estimation based on EM algorithm
  - For DDHMM: for each data sequence $O_l=\{o_{l1},o_{l2},\ldots,o_{lT}\}$, treat its state sequence $S_l=\{s_{l1},\ldots,s_{lT}\}$ as missing data.
  - For Gaussian mixture CDHMM: treat both state sequence S_l and mixture component label sequence $l_l=\{l_{l1},\ldots,l_{lT}\}$ as missing data.

Baum-Welch algorithm: DDHMM(I)

- E-step:

$$Q(\Lambda;\Lambda^{(n)}) = E_{\{S_l\}}\left[\ln p(O_1,\cdots O_L,S_1,\cdots,S_L\mid\Lambda)\ \mid O_1,\cdots O_L\ ,\Lambda^{(n)}\right]$$

$$=\sum_{S_1\cdots S_L}\left[\sum_{l=1}^{L}\ln p(O_l,S_l\mid\Lambda)\right]\cdot\prod_{l=1}^{L}p(S_l\mid O_l,\Lambda^{(n)})=\sum_{l=1}^{L}\sum_{S_l}\ln p(O_l,S_l\mid\Lambda)\cdot p(S_l\mid O_l,\Lambda^{(n)})$$

$$=\sum_{l=1}^{L}\sum_{s_{l1}\cdots s_{lT_l}}\left[\ln\pi_{s_{l1}}+\cdot\ln b_{s_{l1}}(o_{l1})+\sum_{t=2}^{T_l}\ln a_{s_{t-1}s_t}+\sum_{t=2}^{T_l}\ln b_{s_t}(o_t)\right]\cdot p(S_l\mid O_l,\Lambda^{(n)})$$

$$=\sum_{l=1}^{L}\sum_{i=1}^{N}\ln\pi_i\cdot\Pr(s_{l1}=s_i\mid O_l,\Lambda^{(n)})+\sum_{l=1}^{L}\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{t=2}^{T_l}\ln a_{ij}\cdot\Pr(s_{lt-1}=s_i,s_{lt}=s_j\mid O_l,\Lambda^{(n)})$$

$$+\sum_{l=1}^{L}\sum_{i=1}^{N}\sum_{m=1}^{M}\sum_{t=1}^{T_l}\ln b_i(v_M)\cdot\Pr(s_{lt}=s_i,o_t=v_m\mid O_l,\Lambda^{(n)})$$

$$=Q(\pi;\pi^{(n)})+Q(A;A^{(n)})+Q(B;B^{(n)})$$

Baum-Welch algorithm: DDHMM(II)

- M-step: constrained maximization.

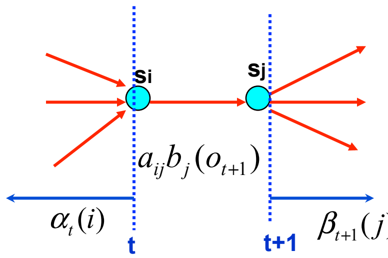
$$\frac{\partial Q(\pi;\pi^{(n)})}{\partial\pi_i}=0\ \Rightarrow\ \pi_i^{(n+1)}=\frac{\sum_{l=1}^{L}\Pr(s_{l1}=s_i\mid O_l,\Lambda^{(n)})}{\sum_{l=1}^{L}\sum_{i=1}^{N}\Pr(s_{l1}=s_i\mid O_l,\Lambda^{(n)})}$$

$$\frac{\partial Q(A;A^{(n)})}{\partial a_{ij}}=0\ \Rightarrow\ a_{ij}^{(n+1)}=\frac{\sum_{l=1}^{L}\sum_{t=2}^{T_l}\Pr(s_{lt-1}=s_i,s_{lt}=s_j\mid O_l,\Lambda^{(n)})}{\sum_{l=1}^{L}\sum_{t=2}^{T_l}\sum_{j=1}^{N}\Pr(s_{lt-1}=s_i,s_{lt}=s_j\mid O_l,\Lambda^{(n)})}=\frac{\sum_{l=1}^{L}\sum_{t=2}^{T_l}\Pr(s_{lt-1}=s_i,s_{lt}=s_j\mid O_l,\Lambda^{(n)})}{\sum_{l=1}^{L}\sum_{t=1}^{T_l-1}\Pr(s_{lt}=s_i\mid O_l,\Lambda^{(n)})}$$

$$\frac{\partial Q(B;B^{(n)})}{\partial b_i(v_m)}=0\ \Rightarrow\ b_i(v_m)=\frac{\sum_{l=1}^{L}\sum_{t=1}^{T_l}\Pr(s_t=s_i,o_t=v_m\mid O_l,\Lambda^{(n)})}{\sum_{l=1}^{L}\sum_{t=1}^{T_l}\sum_{m=1}^{M}\Pr(s_t=s_i,o_t=v_m\mid O_l,\Lambda^{(n)})}=\frac{\sum_{l=1}^{L}\sum_{t=1}^{T_l}\Pr(s_t=s_i,o_t=v_m\mid O_l,\Lambda^{(n)})}{\sum_{l=1}^{L}\sum_{t=1}^{T_l}\Pr(s_t=s_i\mid O_l,\Lambda^{(n)})}$$

Baum-Welch algorithm: DDHMM(III)

- How to calculate the posteriori probabilities?



$$\Pr(s_t = s_i, s_{t+1} = s_j \mid O_l, \Lambda^{(n)}) = \frac{\sum \text{prob of all paths passing } s_i \text{ at } t \text{ and } s_j \text{ at } t+1}{\sum \text{prob of all paths}}$$

$$= \frac{\alpha_t(i) \cdot a_{ij} b_j(o_{t+1}) \cdot \beta_{t+1}(j)}{\Pr(O_l \mid \Lambda^{(n)})} = \frac{\alpha_t(i) \cdot a_{ij} b_j(o_{t+1}) \cdot \beta_{t+1}(j)}{\sum_{i=1}^{N} \alpha_{T_l}(i)} = \frac{\alpha_t(i) \cdot a_{ij} b_j(o_{t+1}) \cdot \beta_{t+1}(j)}{P_l}$$

$$\equiv \xi_t^{(l)}(i,j)$$

Baum-Welch algorithm: DDHMM(IV)

- Final results: one iteration, from $\Lambda^{(n)} = \{A^{(n)}, B^{(n)}, \pi^{(n)}\}$

$$\pi_i^{(n+1)} = \frac{\sum_{l=1}^{L} \Pr(s_{l1} = s_i \mid O_l, \Lambda^{(n)})}{\sum_{l=1}^{L} \sum_{i=1}^{N} \Pr(s_{l1} = s_i \mid O_l, \Lambda^{(n)})} = \sum_{l=1}^{L} \sum_{j=1}^{N} \xi_1^{(l)}(i,j)$$

$$a_{ij}^{(n+1)} = \frac{\sum_{l=1}^{L} \sum_{t=2}^{T_l} \Pr(s_{lt-1} = s_i, s_{lt} = s_j \mid O_l, \Lambda^{(n)})}{\sum_{l=1}^{L} \sum_{t=2}^{T_l} \sum_{j=1}^{N} \Pr(s_{lt-1} = s_i, s_{lt} = s_j \mid O_l, \Lambda^{(n)})} = \frac{\sum_{l=1}^{L} \sum_{t=2}^{T_l} \xi_{t-1}^{(l)}(i,j)}{\sum_{l=1}^{L} \sum_{t=2}^{T_l} \sum_{j=1}^{N} \xi_{t-1}^{(l)}(i,j)}$$

$$b_i(v_m) = \frac{\sum_{l=1}^{L} \sum_{t=1}^{T_l} \Pr(s_t = s_i, o_t = v_m \mid O_l, \Lambda^{(n)})}{\sum_{l=1}^{L} \sum_{t=1}^{T_l} \sum_{m=1}^{M} \Pr(s_t = s_i, o_t = v_m \mid O_l, \Lambda^{(n)})} = \frac{\sum_{l=1}^{L} \sum_{t=1}^{T_l} \sum_{j=1}^{N} \xi_t^{(l)}(i,j) \cdot \delta(o_{lt} - v_m)}{\sum_{l=1}^{L} \sum_{t=1}^{T_l} \sum_{j=1}^{N} \xi_t^{(l)}(i,j)}$$

Baum-Welch algorithm: Gaussian mixture CDHMM(I)

- Treat both state sequence S_l and mixture component label sequence l_l as missing data.
- Only B estimation is different.
- E-step:

$$Q(B;B^{(n)}) = \sum_{l=1}^{L}\sum_{i=1}^{N}\sum_{k=1}^{K}\sum_{t=1}^{T_l} \ln b_{ik}(X_{lt}) \cdot \Pr(s_{lt}=s_i, l_{lt}=k \mid O_l, \Lambda^{(n)})$$

$$= \sum_{l=1}^{L}\sum_{i=1}^{N}\sum_{k=1}^{K}\sum_{t=1}^{T_l} \left[\ln \omega_{ik} - \frac{n}{2}\ln|\Sigma_{ik}| - \frac{1}{2}\cdot(X_{lt}-\mu_{ik})^t \Sigma_{ik}^{-1}(X_{lt}-\mu_{ik}) \right] \cdot \Pr(s_{lt}=s_i, l_{lt}=k \mid O_l, \Lambda^{(n)})$$

- M-step: unconstrained maximization

$$\frac{\partial Q(B;B^{(n)})}{\partial \mu_{ik}} = 0 \Rightarrow \mu_{ik}^{(n+1)} = \frac{\sum_{l=1}^{L}\sum_{t=1}^{L_t} X_{lt} \cdot \Pr(s_{lt}=s_i, l_{lt}=k \mid O_l, \Lambda^{(n)})}{\sum_{l=1}^{L}\sum_{t=1}^{L_t} \Pr(s_{lt}=s_i, l_{lt}=k \mid O_l, \Lambda^{(n)})}$$

Baum-Welch algorithm: Gaussian mixture CDHMM(II)

$$\frac{\partial Q(B;B^{(n)})}{\partial \Sigma_{ik}} = 0 \Rightarrow \Sigma_{ik}^{(n+1)} = \frac{\sum_{l=1}^{L}\sum_{t=1}^{L_t} (X_{lt}-\mu_{ik}^{(n)})^t \cdot (X_{lt}-\mu_{ik}^{(n)}) \cdot \Pr(s_{lt}=s_i, l_{lt}=k \mid O_l, \Lambda^{(n)})}{\sum_{l=1}^{L}\sum_{t=1}^{L_t} \Pr(s_{lt}=s_i, l_{lt}=k \mid O_l, \Lambda^{(n)})}$$

$$\frac{\partial Q(B;B^{(n)})}{\partial \omega_{ik}} = 0 \Rightarrow \omega_{ik}^{(n+1)} = \frac{\sum_{l=1}^{L}\sum_{t=1}^{L_t} \Pr(s_{lt}=s_i, l_{lt}=k \mid O_l, \Lambda^{(n)})}{\sum_{l=1}^{L}\sum_{t=1}^{L_t}\sum_{k=1}^{K} \Pr(s_{lt}=s_i, l_{lt}=k \mid O_l, \Lambda^{(n)})}$$

where the posteriori probabilities are calculated as:

$$\Pr(s_{lt}=s_i, l_{lt}=k \mid O_l, \Lambda^{(n)}) \equiv \zeta_t^{(l)}(i,k) = \frac{\alpha_t^{(l)}(i)\cdot\beta_t^{(l)}(i)\cdot\gamma_{ik}^{(l)}(t)}{P_l}$$

$$\text{where } \gamma_{ik}^{(l)}(t) = \frac{\omega_{ik}^{(n)} \cdot N(X_{lt} \mid \mu_{ik}^{(n)}, \Sigma_{ik}^{(n)})}{\sum_{k=1}^{K} \omega_{ik}^{(n)} \cdot N(X_{lt} \mid \mu_{ik}^{(n)}, \Sigma_{ik}^{(n)})}$$

HMM Training: summary

- For HMM model $\Lambda = \{A, B, \pi\}$ and a training data set *D = {O1, O2, …, OL}*,
  1. Initialization $\Lambda^{(0)} = \{A^{(0)}, B^{(0)}, \pi^{(0)}\}$, set *n=0 ;*
  2. For each parameter to be estimated, declare and initialize two accumulator variables (one for numerator, another for denominator in updating formula).
  3. For each observation sequence O_l *(l=1,2,…,L)*:
     a) Calculate $\alpha_t(i)$ and $\beta_t(i)$ based on $\Lambda^{(n)}$.
     b) Calculate all other posteriori probabilities
     c) Accumulate the numerator and denominator accumulators for each HMM parameter.
  4. HMM parameters update: $\Lambda^{(n+1)} =$ the numerators divided by the denominators.
  5. *n=n+1*; Go to step 2 until convergence.

HMM implementation Issues

- Logarithm representation for all HMM parameters
  - To avoid underflow in computer multiplication.
  - Re-write all computation formula based on:

$$c = ab \Rightarrow \log c = \log(ab) = \log a + \log b$$

$$e = a/b \Rightarrow \log e = \log(a/b) = \log a - \log b$$

$$d = a \pm b \Rightarrow \log d = \log(a \pm b) = \log a + \log[1 \pm \exp(\log b - \log a)]$$

- HMM topology: case-dependent; In speech recognition, always left-right Gaussian mixture CDHMM.
  - For a phoneme model: 3-state left-right HMM.
  - For a whole word (English digit): 10-state left-right HMM.
  - Gaussian mixture number per state: depends on amount of data.
- HMM initialization:
  - A bad initial values → bad local maximum
  - For CDHMM: in HTK toolkit
    - Uniform segmentation →VQ (LBG/K-means) → Gaussian
    - Flat start